AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

Claims 1-11 (Cancelled)

Claim 12 (Currently Amended): A ~~method of~~ system for elliptic curve encryption, the system comprising a computer having a computer readable medium having stored thereon instructions which, when executed by a processor of the computer, causes the processor to perform the steps of:

(a) selecting an elliptic curve E_p (a,b) of the form $y^2=x^3 + ax + b \bmod (p)$, wherein p is a prime number, wherein a and b are non-negative integers less than p satisfying the formula $4a^3 + 27b^2 \bmod (p)$ not equal to 0;

(b) generating a ~~large~~ 160 bit random number by a method of concatenation of a number of smaller random numbers;

(c) generating a well hidden point G (x,y) on the elliptic curve E_p (a,b) by scalar multiplication of a point B (x,y) on the elliptic curve with a ~~large~~ random integer ~~which~~ M further ~~comprises~~ comprising the steps of:

(i) converting the ~~large~~ random integer M into a series of powers of 2^{31};

(ii) converting each coefficient of 2^{31} obtained from above step into a binary series;

(iii) ~~multiplication of~~ multiplying the binary series obtained from steps (i) and (ii) above with the point B (x,y) on the elliptic curve;

(d) generating a private key n_A [[(of about >=]] greater than or equal to 160 bits ~~length)~~;

(e) generating ~~of~~ a public key $P_A(x,y)$ given by the formula $P_A(x,y) = (n_A \cdot G(x,y)) \bmod (p)$;

(f) encrypting ~~the~~ an input message MSG further comprising the steps of:

(i) generating a ~~large~~ random integer K;

(ii) setting $P_{mask}(x,y) = k \cdot P_A(x,y) \bmod (p)$;

(iii) setting $P_k(x,y) = k \cdot G(x,y)$ mod (p);

(iv) accepting the input message MSG to be encrypted;

(v) converting the input message into a point $P_c(x,y)$;

(vi) generating a random point $P_m(x,y)$ on the elliptic curve $E_p(a,b)$;

(vii) setting $P_e(x,y) = (P_c(x,y) - P_m(x,y))$;

(viii) setting $P_{mk}(x,y) = (P_m(x,y) + P_{mask}(x,y))$ mod (p);

(ix) returning $P_k(x)$, $P_e(x,y)$, and $P_{mk}(x)$ as a ciphered text; and

(g) decrypting the ciphered text further comprising the steps of:

(i) getting the ciphered text ($P_k(x)$, $P_a(x,y)$, and P_{mk} (x);

(ii) computing $P_k(y)$ from $P_k(x)$ using the elliptic curve $E_p(a,b)$;

(iii) computing $P_{mk}(y)$ from $P_{mk}(x)$ using elliptic curve $E_p(a,b)$;

(iv) computing $P_{ak}(x,y) = (n_{A\cdot}\ P_k(x,y))$ mod (p);

(v) computing $P_m(x,y) = P_{mk}(x,y) - P_{ak}(x,y))$ mod (p);

(vi) computing $P_c(x,y) = P_m(x,y) + P_e(x,y)$;

(vii) converting $P_c(x,y)$ into the input message MSG.

Claim 13 (Currently Amended): The ~~method of~~ system for elliptic curve encryption as claimed in claim 12, wherein the ~~said~~ number p appearing in selection of elliptic curve is ~~about~~ a 160 bit length prime number.

Claim 14 (Currently Amended): The ~~method of~~ system for elliptic curve encryption as claimed in claim 12, wherein the said method of generating ~~any large~~ the random integer M comprises the steps of:

(i) setting a variable I [[=]] equal to 0;

(ii) setting M to null;

(iii) determining whether I[[<]] is less than 6;

(iv) going to ~~next~~ step (vi) if ~~true~~ I is less than 6;

(v) returning M as a result if ~~false~~ I is not less than 6;

(vi) generating a random number RI within (0,1);

(vii) multiplying RI with 10^9 to obtain variable BINT [[-]],wherein BINT is an integer ~~of size~~ having 9 digits;

(viii) concatenating BINT to M;

(ix) setting I [[=]] equal to I + 1;

(x) returning to step (iii).

Claim 15 (Currently Amended): The ~~method of~~ system for elliptic curve encryption as claimed in claim 12, wherein said conversion of the ~~large~~ random integer into a series of powers of 2^{31} and conversion of each coefficient m_n of said 2^{31} series thus obtained for scalar multiplication for said random integer with the said point B(x,y) on said elliptic curve E_p (a,b) comprises the steps of:

(i) accepting ~~a big~~ the integer M;

(ii) setting a variable T31 equal to 2^{31};

(iii) setting a variable LIM [[=]] equal to a size of M [[(]]in bits[[)]] and initializing an array A() with size LIM;

(iv) setting a variable INCRE equal to [[zero]] 0;

(v) setting a variable N equal to M modulus T31;

(vi) setting M [[=]] equal to INT(M/T31);

(vii) determining whether N is equal to 0;

(viii) going to ~~next~~ step (x) if ~~true~~ N is equal to 0;

(ix) going to step (xxiv) if ~~false~~ N is not equal to 0;

(x) determining whether M is equal to 0;

(xi) going to ~~next~~ step (xiii) if ~~true~~ M is equal to 0;

(xii) going to step (xxvi) if ~~false~~ M is not equal to 0;

(xiii) setting I [[=]] equal to 0 and J [[=]] equal to 0;

(xiv) determining whether I [[≥]] is greater than or equal to LIM;

(xv) going to ~~next~~ step (xvii) if ~~false~~ I is not greater than or equal to LIM;

(xvi) going to step (xxviii) if ~~true~~ I is greater than or equal to LIM;

(xvii) determining whether A(I) is equal to 1;

(xviii) going to ~~next~~ step (xx) if ~~true~~ A(I) is equal to 1;

(xix) returning to step (xxii) if ~~false~~ A(I) is not equal to 1;

(xx) setting B (J) [[=]] equal to I;

(xxi) incrementing J by 1;

(xxii) incrementing I by 1;

(xxiii) returning to step (xiv);

(xxiv) calling a function BSERIES (N, INCRE) and updating array A();

(xxv) returning to step (x);

(xxvi) setting a variable INCRE [[=]] equal to INCRE + 31;

(xxvii) returning to step (v);

(xxviii) returning array B() as a result.

Claim 16 (Currently Amended): The ~~method of~~ system for elliptic curve encryption as claimed in claim 15, wherein said conversion of the ~~large~~ random integer into a series of powers of 2^{31} and said conversion of each coefficient m_n of said 2^{31} series thus obtained for the said scalar multiplication of the said random integer with the said point B(x,y) on said elliptic curve E_p (a,b) further comprises the steps of:

(i) accepting N and INCRE;

(ii) assigning an array BARRAY as an array of values ~~which~~ that are powers of 2([2^0,........2^{30}]);

(iii) setting a variable SIZE [[=]] equal to size of N ~~(in digits)~~;

(iv) computing a POINTER [[=]], wherein the POINTER is equal to 3·(SIZE)+INT(SIZE/3)-4;

(v) determining whether the POINTER [[<]] is less than 2;

(vi) going to ~~next~~ step (viii) if ~~true~~ the POINTER is less than 2;

(vii) going to step (ix) if ~~false~~ the POINTER is not less than 2;

(viii) setting the POINTER equal to [[zero]] 0;

(ix) determining whether [[(]]BARRAY(POINTER) [[≥]] is greater than or equal to N[[)]];

(x) going to ~~next~~ step (xii) if ~~true~~ BARRAY(POINTER) is greater than or equal to N;

(xi) going to step (xx) if ~~false~~ BARRAY(POINTER) is not greater than or equal to N;

(xii) determining whether BARRAY (POINTER)[[=]] is equal to N;

(xiii) going to ~~next~~ step (xv) if ~~true~~ BARRAY (POINTER) is equal to N;

(xiv) going to step (xvii) if ~~false~~ BARRAY (POINTER) is not equal to N;

(xv) setting A (POINTER + INCRE) equal to 1;

(xvi) returning array A () as a result;

(xvii) setting A ((POINTER - 1) + INCRE) equal to 1;

(xviii) computing N[[=]], wherein N is equal to N-BARRAY(POINTER-1);

(xix) returning to step (iii);

(xx) setting <u>the</u> POINTER [[=]] <u>equal to</u> POINTER + 1;

(xxi) returning to step (ix).

Claim 17 (Currently Amended): The ~~method of~~ <u>system for</u> elliptic curve encryption as claimed in claim 16, wherein said scalar multiplication of the said binary series with the said point B(x,y) on the said elliptic curve $E_p(a,b)$ comprises the steps of:

(i) accepting B(x,y), <u>wherein B(x,y) is</u> a point on $E_p(a,b)$;

(ii) accepting <u>an</u> array B() with size LIM;

(iii) setting <u>another variable</u> I [[=]] <u>equal to</u> 0 and <u>another variable</u> J [[=]] <u>equal to</u> 0;

(iv) determining whether B(J)[[=]] <u>is equal to</u> I;

(v) going to ~~next~~ step <u>(vii)</u> if ~~true~~ <u>B(J) is equal to I</u>;

(vi) going to step (xxv) if ~~false~~ <u>B(J) is not equal to I</u>;

(vii) setting PARR (x,y) (J) equal to B(x,y);

(viii) incrementing J by 1;

(ix) determining whether J is equal to LIM;

(x) going to ~~next~~ step <u>(xii)</u> if ~~true~~ <u>J is equal to LIM</u>;

(xi) going to step (xxv) if ~~false~~ <u>J is not equal to LIM</u>;

(xii) setting K [[=]] <u>equal to</u> [[zero]] <u>0</u>;

(xiii) determining whether K[[>]] <u>is greater than</u> 0;

(xiv) going to ~~next~~ step <u>(xvi)</u> if ~~true~~ <u>K is greater than 0</u>;

(xv) going to step (xxii) if ~~false~~ <u>K is not greater than 0</u>;

(xvi) computing FP(x,y)[[=]]<u>, wherein FP(x,y) is equal to</u> FP(x,y) + PARR(x,y) (K);

(xvii) incrementing K by 1;

(xviii) determining whether K[[=]] <u>is equal to</u> LIM;

(xix) going to ~~next~~ <u>step (xxi)</u> if ~~true~~ <u>K is equal to LIM</u>;

(xx) returning to step (xiii) if ~~false~~ <u>K is not equal to LIM</u>;

(xxi) returning FP(x,y) as <u>a</u> result;

(xxii) setting FP(x,y) equal to PARR(x,y) (K);

(xxiii) incrementing K by 1;

(xxiv) returning to step (xiii);

(xxv) incrementing I by 1;

(xxvi) setting B(x,y) [[=]] equal to B(x,y) + B(x,y);

(xxvii) returning to step (iv).

Claim 18 (Currently Amended): The ~~method~~ system for of elliptic curve encryption as claimed in claim 12, wherein said public key $P_A(x,y)$ is also a point on said elliptic curve $E_p(a,b)$.

Claim 19 (Currently Amended): The ~~method~~ system for of elliptic curve encryption as claimed in claim 12, wherein the generation of said private key n_A and said public key $P_A(x,y)$ comprises the steps of:

(i) entering a ~~big~~ odd integer p ~~of size ≥~~ greater than or equal to 160 bits;

(ii) determining whether p is a prime number;

(iii) going to ~~next~~ step (v) if p is a prime number;

(iv) going to step (xix) if p is not a prime number;

(v) entering an ~~small~~ integer a [[>]], wherein a is greater than 0;

(vi) setting an integer b [[=]] equal to 0 and a variable x [[=]] equal to 1;

(vii) determining whether $4a^3 + 27b^2$ mod (p) [[= zero]] is equal to 0;

(viii) going to ~~next~~ step (x) if ~~false~~ $4a^3 + 27b^2$ mod is not equal to 0;

(ix) incrementing b by 1 if ~~true~~ $4a^3 + 27b^2$ mod (p) is equal to 0 and going to step (vii);

(x) setting z [[$(=y^2)$ =]] equal to $x^3 + ax + b$, wherein z is y^2;

(xi) determining whether z[[$(=y^2)$]] is a perfect square;

(xii) going to step (xxi) if z is not a perfect square;

(xiii) setting B(x,y) equal to (x,y) if z is a perfect square;

(xiv) selecting a ~~large~~ random integer r_1;

(xv) setting G(x,y) [[=]] equal to (r_1 B(x,y)) mod (p);

(xvi) selecting a ~~large~~ random integer n_A;

(xvii) setting $P_A(x,y)$ [[=]] equal to ($n_A \cdot G(x,y)$) mod (p);

(xviii) returning $P_A(x,y)$ as a public key and returning n_A as a private key;

(xix) incrementing p by 2;

(xx) returning to step (ii);

(xxi) incrementing x by 1;

(xxii) determining whether x [[>]] is greater than 900;

(xxiii) going to ~~next~~ step <u>(xxv)</u> if ~~true~~ <u>x is greater than 900</u>;

(xxiv) going to step (x) if ~~false~~ <u>x is not greater than 900</u>;

(xxv) incrementing b by 1;

(xxvi) setting x [[=]] <u>equal to</u> 1;

(xxvii) returning to step (vii).

Claim 20-21 (Cancelled)